Mobility Support in Unified Communication Networks

Helen J. Wang and Randy H. Katz
{helenjw, randy}@cs.berkeley.edu
Computer Science Department
University of California, Berkeley

January 31, 2001

Abstract

Rapid advances in communication networks and device technologies have pushed the market demand for unified communications across heterogeneous networks and devices. To meet this demand, the research community and communication industry is experimenting and building Internet-based, unified communication network systems (UCN) in which heterogeneous devices are used in an integrated fashion and access networks are linked together through a core IP network. In this paper, we investigate the mobility issues in UCN systems. We discuss the new meanings of the traditional mobility issues such as personal mobility and terminal mobility, and present our design, analysis, implementation of a new type of mobility, *service mobility* where active services can be retained across heterogeneous devices and networks.

1 Introduction

Today, we have seen expeditious progress in human communication mechanisms (such as PDAs, cellular phones, pagers, and instant messaging) as well as a proliferation of communication networks, including the Public Switched Telephone Network (PSTN), paging networks, wireless networks such as Metricom[14] and OmniSky[16], and cellular systems with various air interface technologies and standards including the upcoming Third Generation WCDMA systems. It is commonplace that people communicate with different devices at various occasions, and can be reached at multiple communication endpoints. This phenomenon has spurred a demand for unified communication services that integrate one's various communication mechanisms in a meaningful way. For example, the unified messaging service, a popular commercial service today, unifies one's e-mail and voicemail into a common in-box accessible from heterogeneous devices. In addition, forward-looking, industry players have begun to acknowledge the user needs for more generalized unified communications including call management and call control functions [19]. Driven by the market, unified communication network (UCN) system prototypes are emerging from both the commercial world and the research community (such as the ICEBERG [21] and SIP-based systems [18]). As telecommunications networks are migrating towards Internet technology, with voice over IP maturing rapidly, UCNs are mostly built on top of the Internet leveraging its support for heterogeneity. Figure 1 gives an overview of a generic Internet-based UCN system. Terminal agents (TAs) perform communication establishment and maintenance on behalf of communication endpoints. In these systems, the key architectural components (such as the TAs in the figure) and protocols for communication unification form an overlay network on top of the Internet; and the heterogeneous communication networks are bridged into the IP network through gateways performing signaling translations and data stream packetization and transformation.

In this paper, we examine the mobility issues in UCN systems, which have not been covered in the

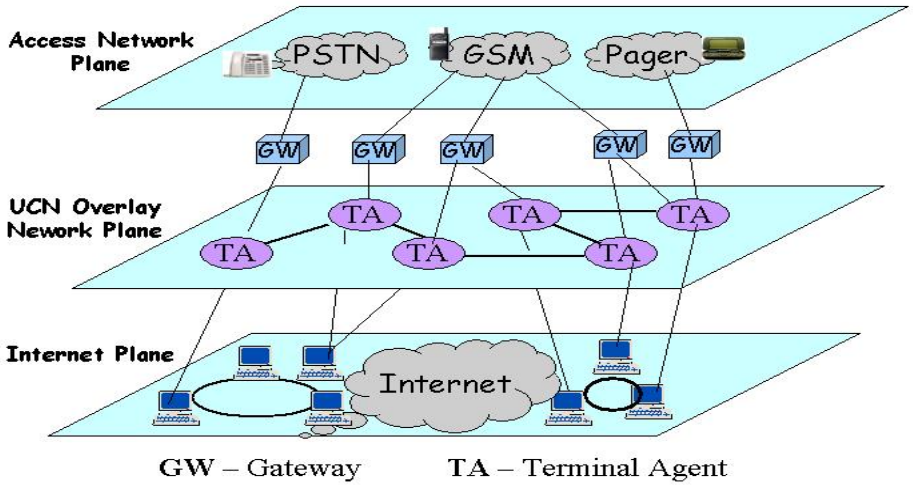

Figure 1: A Unified Communication System

literature. For traditional mobility problems such as *personal mobility* and *terminal mobility*, we discuss their new implications in UCN systems (Section 2). Most of our paper is devoted to the design and analysis for supporting *service mobility*, a new type of mobility entailed by UCN (Section 2 and Section 3). Service mobility support seemed simple at first sight, but designing a *correct* protocol taking care of all possible communication session dynamics such as agent fault recovery or simultaneous actions is tricky. To this end, we had to resort to designing a new application-level lightweight group service to support efficient service mobility for correctness, and no existing mechanisms are sufficient (Section 3.1). We have successfully implemented our mobility design in our UCN test-bed. We present our implementation experience and service mobility performance in Section 4, and conclude in Section 5. The related work is discussed throughout the paper.

2 Mobility Issues in UCN

Mobility refers to the ability of retaining session-based services (such as communication services) in face of any changes, such as location change (Terminal Mobility), role change (Personal Mobility), or changes in devices in use (Service Mobility). In this section, we discuss the new implications in traditional mobility issues such as *terminal mobility* and *personal mobility*, we will also introduce *service mobility*, a new type of mobility entailed by the UCN systems.

2.1 Terminal Mobility

Terminal Mobility refers to the ability of a terminal to remain connected to the network while moving from one location to another without any service disruption. Mobile IP [15] tackles the terminal mobility problem of a mobile host moving from one IP network to another. Cellular networks also allow mobile phones to move from one cell to another or even across different administrative domains through the "handoff" operation. The key solution to the ter-

minal mobility problem is to dissociate a terminal's identity from its physical location, and introduce a level of indirection in routing for receiving and forwarding data to the mobile terminal at its present location.

Although the IP network and cellular networks effectively tackled the terminal mobility problem, for cost and resource-efficient communication, UCN systems still need to provide terminal mobility support. Imagine a user talking on her cell phone is driving a long way, handing off across many cells, roaming into a new service provider domain, and getting farther away from the gateway (between her cellular network and IP-core UCN network). It would be more desirable to use a nearby gateway to serve the communication for a lower cost and more efficient use of cellular network resources and IP network resources.

We call the operation of switching from one gateway to another *gateway handoff*. To enable gateway handoff, additional APIs must be added into the gateway design so that access networks can notify the UCN of the terminal mobility information, then the UCN (actually the serving terminal agent) selects and switches to a better gateway for the communication. Gateway selection can be done through the gateway location protocol proposed by the IETF [17] or wide-area service discovery protocol [5]. For seamless gateway handoff, the terminal agent does not disconnect the old gateway before the new gateway obtains all the necessary state. In addition, the system must not perform gateway handoff too frequently for system stability.

2.2 Personal Mobility

Personal mobility refers to the idea that the person is the communication endpoint rather than the various devices the person owns. This concept originates from the one-number service of the Personal Communication Systems (PCS) [22], where a person is identified by one telephone number only, instead of her home number, office number, etc.. Personal mobility is at the heart of unifying various devices in UCN. Similar to the PCS systems, a globally unique ID is required for any individual. A name mapping service is needed for mapping domain- or device-specific IDs to user's unique IDs. Incoming communications to a user are redirected to specific devices by user preferences or system defaults.

2.3 Service Mobility

Service Mobility is a new type of mobility required in UCN systems. As people own multiple devices, they may desire to switch from one device to another (and as a result, switching from one network to another) in the middle of a conversation. As an example, Alice talking on her desktop phone in her office may have to leave for another appointment, and would like to switch from her desktop phone to her cell-phone without hanging up the current communication and initiating a new one from her cellphone. We call the operation of switching among the devices *service handoff*, and the ability of retaining access to services across different devices and networks *service mobility*.

In telecommunications, the term "service mobility" usually means "service portability", the ability for diverse networks to share user service profiles and to carry out the same set of services in each network [6]. However, when one crosses network boundaries during an active service session, that service is terminated. The user must then restart the service in the new network. In contrast, service mobility provides *seamless* integration of heterogeneous devices from diverse networks, as if they are accessing the same network. The concept of *service mobility* originally comes from [12]; it is also included as one of the Session Initiation Protocol mobility issues [20]. However, the literature contains no detailed design or complete solutions for service mobility, which is the central theme of this paper.

In the next section, we present our design rationale as well as detailed design and analysis for service mobility.

3 Supporting Service Mobility

Service handoff can be completely user-driven with a small set of system primitives: the user invites the new device into the conversation, then hangs up the

previous device. Automatic service handoff can be enabled with the same set of system primitives plus other context aware or tracking components, such as a component that checks the availability of all devices, and keeps track of the "best" device to use. "Best" could be defined through user preferences plus some system defaults, and is the policy on when and what device to service handoff. In this paper, we will not investigate further on policy issues for service handoff, but only on the basic service handoff mechanisms.

A naive solution [20] to service handoff is simply to augment the signaling protocol (the control protocol for communication establishment and maintenance) by allowing a user to invite a new device in the middle of a communication session. In reality, the terminal agent (see Figure 1 in Section 1) accomplishes this invitation on user's behalf, and it sends a notification or registration message to the other terminal agents in the session. The notification message includes the data path information of the new endpoint such as the IP address and port number on which it receives the data stream so that other endpoints can establish data flow with the new endpoint.

However, such a simple service handoff protocol is not sufficient. It fails when communication participants perform handoffs simultaneously, because the terminal agents for all the new endpoints in the new session do not know one another's existence, and therefore cannot send the notification message to one another, which jeopardizes the communication session. The same problem happens if terminal agent failure recovery occurs during service handoff, where the recovered agent has a different IP address.

The tricky problem here is how to send the notification messages to all involved terminal agents of the session in face of all possible session dynamics such as simultaneous handoffs and agent failure recoveries. One approach is to have one centralized agent to keep track of the membership of the terminal agents (this is similar to the use of Multipoint Control Unit (MCU) for bridged conferences in H.323 [8] and SIP [18] signaling protocols). However, this centralized approach yields a single point of failure and bottleneck in the control path.

As another approach, we could push the terminal agent membership problem down one level by using IP multicast. New endpoint notifications are sent to a shared, IP multicast channel which serves as a level of indirection hiding the identity and the location of the terminal agents and copes with the dynamic membership of the terminal agents. Although the group service provided by IP multicast is desirable here, with one multicast group per conversation, IP multicast scaling issues arise, namely IP address scarcity and the router state scalability. In addition, our group size tends to be small, and the information communicated among the group members is also small. Thus, we do not leverage the efficient transport offered by multicast. The practical deployment problems of IP multicast also prevent us from relying on it.

What we really need here is an application-level light-weight group service that has equivalent functionalities of the IP multicast group service, but does not suffer from the scaling problems of IP multicast. We considered using existing application-level multicast protocols such as Narada [3], Overcast [11], ScatterCast [2] or Yallcast [7]. They emphasize efficient multi-party transport and require each member to maintain the topology of the group. All these protocols are too heavy-weight for our light traffic control signaling among a small group of terminal agents.

As a result, we designed a stand-alone application-level lightweight group service package (ALGS) ourselves, and tailored its design for our needs. We first describe how ALGS is used for service handoff, then present our design for ALGS in detail (Section 3.1).

A group (communication session) is identified by the concatenation of the call initator's ID and the calling time. The call initiator's terminal agent creates the group. Invitations to join the call include the group ID as well as the group membership list known to the inviter. ALGS maintains the group membership. Sending a message to a group triggers ALGS to unicast the message to all group members. When a user initiates a service handoff, her terminal agent invites the new endpoint, then the new endpoint's terminal agent sends the new endpoint notification message to all other terminal agents participating the session. Later, the user hangs up the old device, completing the service handoff.

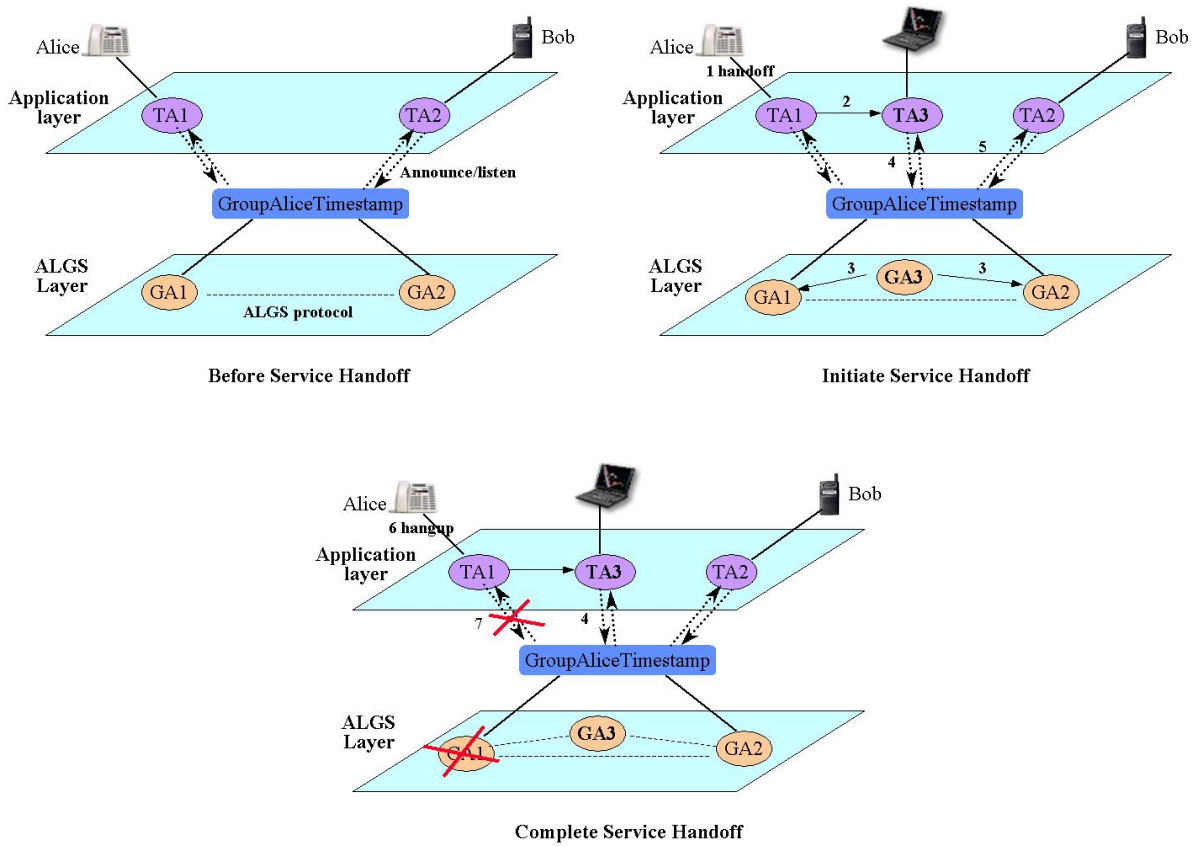

Figure 2: Service handoff through Application-level Lightweight Group Service

What if the notification or hang-up messages get lost for any of the terminal agents? We could let the receiving terminal agent send ACKs back to the sending terminal agent. However, for a session involving multiple parties, such an ARQ-based scheme will cause ACK implosion at the sending terminal agent. In addition, if a network partition separates a terminal agent from others, how do the other terminal agents detect the situation and deallocate resources accordingly? To resolve these issues, we take the announce/listen approach in which each terminal agent periodically announces to the group its endpoint information as heartbeats; this ensures the reliability of the messsages. Each terminal agent also listens to the session keeping track of the liveness of other terminal agents. Not receiving announcements from any particular terminal agent results in a timeout, and a deallocation of the corresponding communication resources. The use of announce/listen greatly enhances the robustness of the communication session including the service handoff operations that may occur. A desirable property of the announce/listen model is that component failures are tolerated as the normal mode of operation, rather than addressed

through a separate recovery procedure [1]: recovery is enabled by simply listening to channel announcements. The announce/listen model initially appeared in IGMP [4], and was further developed and clarified in systems such as the MBone Session Announcement Protocol [13]. Using announce/listen for session maintenance is necessary not only for robust service handoff operation, but also for the reliability and robustness of the general signaling among terminal agents such as signaling messages indicating new device status, or data path endpoint address change due to mobility.

Figure 2 depicts the service handoff process using ALGS. The service handoff process follows these steps:

1. Alice initiates the service handoff by pressing some keys on her telephone to indicate the intent and the target of a service handoff (e.g., "*SHIP" – Service Handoff to IP phone). The serving PSTN gateway intercepts the Dual Tone Multi-Frequency (DTMF) message and relays it to the serving terminal agent (TA1) for Alice's telephone.

2. The terminal agent (TA1) looks up the handoff target endpoint information (IP phone's IP address and port number in this case). Then, TA1 invites a new terminal agent (TA3) which serves the new endpoint (the IP Phone) to join in the conversation session.

3. TA3 creates a new group agent (GA3) for ALGS protocol participation. GA3 joins the group by sending join messages to GA1 and GA2 who represent TA1 and TA2, respectively.

4. TA3 starts announcing its endpoint information to the communication group (GroupAliceTimestamp), as well as listening from the other terminal agents.

5. The new endpoint information gets propagated to Bob cell phone's terminal agent (TA2), then the appropriate data path gets created between Alice's IP Phone on her laptop and Bob's cell phone.

MemberID	Timestamp	inGroup (boolean)

Table 2: Individual member info in membership list

6. Alice hangs up her telephone.

7. TA1 obtains the handoff message, stops announcing and listening to the group, and instructs its GA1 to leave the group as well. This completes the service handoff operation.

Next, we present the detailed design and analysis of the group membership protocol in ALGS in detail.

3.1 The Design and Analysis of the Application-level Lightweight Group Service

Lightweight Application-level Group Service (ALGS) is designed as a library package to its users (such as terminal agents). Table 1 shows the APIs ALGS exposed to its users.

The membership list maintained by each member consists of all members that have ever participated the group, even including those who already left. Each member in the list is represented as in Table 2. "inGroup" indicates whether the member is still in the group. If true, the timestamp is the most recent join time of the member, otherwise, it is the most recent departure time. We choose this representation of the membership list for easy conflict resolution. When two membership lists do not agree, we merge the two lists; then for the entries with the same member ID, only keep the one with the most recent timestamp.

We designed ALGS to meet the following goals:

- Quick convergence in case of membership change
- Minimal bandwidth overhead: avoid broadcast whenever possible

For quick convergence, broadcasts are used whenever there is a membership change. However, this is not sufficient to keep all members up-to-date. Figure 3 depicts simultaneous service handoffs or invitations of new call parties. In this scenario, Group

GroupAgent (groupID, memberlist)	Constructor: creating a group agent object initialized with a member list and a unique group ID
join (myID, groupID, timestamp)	Join a group at a certain time, the new member information is broadcast to the group
leave (myID, groupID, timestamp)	Leave a group at a certain time, departure time recorded, and broadcast to the whole group
send (groupID, message)	Send a message to the group
lostMember (groupID, memberID)	lost a member from the group
foundMember (groupID, memberID)	found a new member in the group

Table 1: The API of the Application-level Lightweight Group Service

agents GA1 and GA2 simultaneously invite GA3 and GA4 into the group as a result of simultaneous handoffs or invitations. Along with the invitations are the membership list from GA1 and GA2 respectively. Then, GA3 and GA4 joins the group by unicasting join messages to both GA1 and GA2 according to the membership list in the invitation. The problem is that while existing members in the group obtain the complete membership information, the new members (GA3 and GA4) do not know of each other's existence.

What is needed here is group membership maintenance after broadcasting the membership changes so that all members will eventually obtain the complete membership information. We use a variant of the Name Dropper protocol [9] for this purpose. Name Dropper protocol is a distributed, randomized node discovery algorithm. Each member periodically selects a member randomly from its list, and sends its membership list to that member, which will merge that list with its own. The Name Dropper protocol meets our goals nicely: it quickly converges in $O(log^2 n)$ rounds with high probability where n is the number of nodes involved, and it does not use much bandwidth (one periodic message per node). Name Dropper assumes weakly connected nodes where an added node knows at least one other member in the group. Our situation satisfies this assumption because the invitation of a new terminal agent includes the current group membership. We make two additions to Name Dropper for quicker convergence in common cases:

- Nodes broadcast join and leave messages when joining or leaving the group.
- After merging a received membership list, the receiver checks whether the sender would benefit from learning the receiver's list, and if so a reply is sent. For example, if the received membership is a subset of or conflicts with the receiver's list, the receiver will send its list to the sender.

As a result, we have quicker convergence in most cases (but bounded by Name Dropper's convergence performance in special cases such as many simultaneous joins) at the cost of broadcast bandwidth upon membership change.

Now we address the group member failure semantics. A group member can fail due to broken software, machine crash, persistent network partition, or member left but all the leave messages were dropped. ALGS cannot differentiate among these failures. In the case of persistent network failure, the network partition between two nodes does not imply network partitions else where. We believe the correct approach is to have localized or independent failure detection at each member. In addition, the nature of the ALGS protocol makes it ineffective to detect the

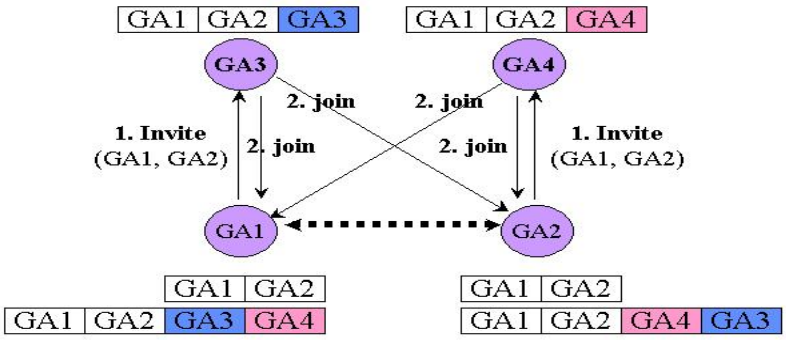

Figure 3: Simultaneous invitation or service handoff.

member loss. So, we decide to push the failure detection to higher layers. The failed member will continue to be listed as in the group at ALGS layer for as long as the group exists. Higher layers, however, may detect the failure via the announce/listen protocol, and can inform the ALGS layer via the lostMember function allowing the ALGS layer to avoid sending wasteful messages to the dead member. If the higher layer learns that the member has recovered, it can inform the ALGS layer via the foundMember function. It is also not fatal if higher layers do not assist failure detection as it only results in a small waste of resources.

Member failure recovery is an orthogonal problem to ALGS. The fault recovery mechanism must save the group ID and current membership list somewhere (either at some endpoint or some cluster state maintainer) for recovered member to continue participating the group membership protocol.

4 Implementation and Performance

We have implemented service mobility and personal mobility in our unified communication system prototype, XXXXXXX. We have not implemented gateway handoff for optimized terminal mobility at the time of writing.

Our test-bed bridges the PSTN and the GSM cellular network into the IP network. Our system components include PSTN and GSM gateways performing signaling and data stream packetization and transformation, a name mapping service mapping domain-specific address to system-wide unique user IDs, a preference registry storing and managing user communication service preferences such as when they wish to be contacted on what device under what circumstances, a personal activity tracking service allowing user to specify what current user behaviors can be tracked, terminal agents that speaks signaling protocol to one another to establish or maintain communication sessions, and a data path creation service which encapsulates data flow establishment and

manipulations. The signaling protocol [X] running among terminal agents includes the session maintenance protocol which performs announce/listen to the session on top of ALGS (described in Section 3.1) to obtain the most up-to-date and complete session state such as membership, data stream information, etc. The soft state nature of the session maintenance protocol assists supporting the service handoff operation and other session dynamics such as agent fault recovery, message losses, and network partitions.

To scale our system to a large user base, we leverage the Ninja vSpace [10], which is a cluster computing platform that provides scalability, failure detection and recovery features to services running on them. We run our components on Ninja vSpace for these desirable features. Thus, our system can be viewed as an overlay network of such clusters with a robust soft state signaling protocol running among the components across the clusters.

We designed and implemented our system to support multi-endpoint service as the basic service instead of the basic two party telephone call model as in telecommunications. The operations that carry out the basic service are the building blocks for additional services. By having multi-endpoint communications as the basic call service, we have enriched the set of the building blocks. The multi-endpoint communication operations are adding a new communication endpoint to the call and removing an existing communication endpoint from the call. Changing a communication endpoint involves adding the new endpoint, and then removing the existing one. These are essentially the mechanisms for our service mobility support. Therefore, service handoff is offered as part of the basic service as well. The service handoff latency is essentially the session setup time, namely, the time it takes for one endpoint to invite another, plus the time for data flow to be established between the new endpoint and the rest of the session. Our measurements show that the local-area (all endpoints use one cluster) session setup time (therefore the service handoff latency) is about 130 milliseconds in average with a standard deviation of 8. We are still conducting wide-area experiments (in which endpoints use different clusters across wide-area) at the time of writing, and will include the wide-area service handoff latency as well as the simultaneous service handoff measurements in the final version of the paper if accepted.

5 Conclusion

In this paper, we have discussed terminal mobility, personal mobility, and service mobility support issues in Internet-based unified communication systems. We presented our detailed design, analysis, and implementation for service mobility. The key challenge in supporting service mobility is the correct service handoff operation in face of simultaneous handoffs, agent failure recoveries, network partitions, or other dynamic conditions. We tackled this by using soft state-based announce-listen protocol over an application-level lightweight group membership protocol. Our analysis showed that the group membership protocol is bandwidth efficient and quick in convergence upon group membership changes. We implemented personal mobility and service mobility support in a unified communication network system prototype, XXXXXXX. We conducted initial measurements of the service handoff latency which is the same as the session setup latency in our system. The latency is low. The extensive study of our system performance is under-way.

References

[1] Elan Amir, Steven McCanne, and Randy H. Katz. An active service framework and its application to real-time multimedia transcoding. In *Proceedings of ACM SIGCOMM 98*. ACM, August 1999. Vancouver, British Columbia.

[2] Y. Chawathe, S. McCanne, and E. Brewer. An architecture for internet content distribution as an infrastructure service. In *Unpublished*, 2000.

[3] Y. Chu, S. G. Rao, and H. Zhang. A case for end system multicas. In *Proceedings of ACM Sigmetrics, Santa Clara, CA*, June 2000.

[4] Steve Deering. *Host Extensions for IP Multicasting*, Aug 1989. IETF RFC-1112.

[5] Steven Czerwinski et.al. An architecture for a secure service discovery service. In *(To Appear) Fifth Annual International Conference on Mobile Computing and Networks*, August 1999.

[6] Nadege Faggion and Cegetel Thierry Hua. Personal communications services through the evolution of fixed and mobile communications and the intelligent network concept. *IEEE Network*, Jul/Aug 1998.

[7] Paul Francis. *Yallcast: Extending the Internet Multicast Architecture*, Sep. 1999.

[8] ITU-T Recommendation H.323. *Packet-Based Multimedia Communications Systems*, feb 2001.

[9] Mor Harchol-Balter, T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *8th Annual ACM-SIGACT/SIGOPS Symposium on Principles of Distributed Computing, Atlanta*, May 1999.

[10] http://www.cs.berkeley.edu/ mikechen/vspace. *The Ninja vSpace Cluster Computing Platform.*

[11] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and Jr J. O'Toole. Overcast: Reliable multicasting with an overlay network. In *OSDI*, 2000.

[12] Anthony D. Joseph, B. R. Badrinath, and Randy H. Katz. The case for services over cascaded networks. In *The First ACM/IEEE International Conference on Wireless and Mobile Multimedia (WoWMoM'98)*, 1998.

[13] Maryann P. Maher and Colin Perkins. *Session Announcement Protocol: Version 2*, nov 1998. IETF Internet Draft: draft-ietf-mmusic-sap-v2-00.txt.

[14] Metricom. http://www.metricom.com/.

[15] Ip routing for wireless/mobile hosts (mobileip). http://www.ietf.org/html.charters/mobileip-charter.html.

[16] Omnisky. http://www.omnisky.com/.

[17] Jonathan Rosenberg and Henning Schulzrinne. Internet telephony gateway location. In *Proceedings of IEEE Infocom 98*. IEEE, August 198.

[18] Henning Schulzrinne. A comprehensive multimedia control architecture for the Internet. In *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video*, May 1997.

[19] Unified communication. http://www.unified-msg.com.

[20] Vakil and et al. *Mobility Management in a SIP Environment Requirements, Functions and Issues. IETF Internet Draft*, December 2000.

[21] Helen J. Wang and et al. Iceberg: An internet-core network architecture for integrated communications. *IEEE Personal Communications*, April 2000.

[22] Mohammed Zaid. Personal mobility in PCS. *IEEE Personal Communications*, Fourth Quarter 1994.